

	CCCCCCCCCCCCC	JJJ	FFFFFFFFFFF	VVV	VVV	444	444
	CCCCCCCCCCCCC	JJJ	FFFFFFFFFFF	VVV	VVV	444	444
	CCCCCCCCCCCCC	JJJ	FFFFFFFFFFF	VVV	VVV	444	444
CCC		JJJ	FFF	VVV	VVV	444	444
CCC		JJJ	FFF	VVV	VVV	444	444
CCC		JJJ	FFF	VVV	VVV	444	444
CCC		JJJ	FFF	VVV	VVV	444	444
CCC		JJJ	FFF	VVV	VVV	444	444
CCC		JJJ	FFF	VVV	VVV	444	444
CCC		JJJ	FFFFFFFFFFF	VVV	VVV	4444444444444444	
CCC		JJJ	FFFFFFFFFFF	VVV	VVV	4444444444444444	
CCC		JJJ	FFFFFFFFFFF	VVV	VVV	4444444444444444	
CCC	JJJ	JJJ	FFF	VVV	VVV		444
CCC	JJJ	JJJ	FFF	VVV	VVV		444
CCC	JJJ	JJJ	FFF	VVV	VVV		444
CCC	JJJ	JJJ	FFF	VVV	VVV		444
CCC	JJJ	JJJ	FFF	VVV	VVV		444
CCC	JJJ	JJJ	FFF	VVV	VVV		444
	CCCCCCCCCCCCC	JJJJJJJJJ	FFF		VVV		444
	CCCCCCCCCCCCC	JJJJJJJJJ	FFF		VVV		444
	CCCCCCCCCCCCC	JJJJJJJJJ	FFF		VVV		444

V03-041 DAS0001 David Solomon 22-Jun-1983
Changes due to implementation of RMS recovery: replace RODBAS
BFILNAM, RFILNAM, JFILNAM, VOLUME, and JNLNAME with FILENAME
and CFILENAME. Remove RODBA JVOLUME. Rename RODBA BDEVICE and
BVOLNAM to be VOLDEVICE and VOLABEL.

V03-040 PRB0197 Paul Beck 1-JUN-1983 15:05
Add RRP\$B_CALL_MSG from CTL\$GB_MSGMASK
Add MESSAG callback code.
Rearrange CJFSV... flags for RECOVER service, and add
CJFSV_FAILOVER and CJFSV_RESTART as well as the RRP equivalents.

V03-039 MKL0075 Mary Kay Lyons 18-May-1983
Add JSB\$V_REMASTER.

V03-038 JSV0266 Joost Verhofstad 18-MAY-1983
Change JSB; use pointers to JNLNAM and ACPNAM buffers
plus add CJF\$C_MXJNLNAML and CJF\$C_MXPRCNAML

V03-037 PRB0183 Paul Beck 2-MAY-1983 19:08:39
Use CSID in RRP. Fix description of JFCB\$C_STRING.
Add TYPE and SUBTYPE fields to RRP.

V03-036 PRB0181 Paul Beck 30-APR-1983
Add RUSYNC bits to RUSS

V03-035 FWM0001 Fred Matthes 26-APR-1983
Add RFSAMPLE code to RODBDEF.

V03-034 PRB0165 Paul Beck 24-APR-1983
Add RCP\$K_DIRECTION request, delete JFCB\$X_ENT_TYPE

V03-033 JSV0211 Joost Verhofstad 06-APR-1983
Change JFCB\$S_DATTIM to 16

V03-032 PRB0148 Paul Beck 31-MAR-1983
Add RCP\$K_MAX_COMMAND

V03-031 JSV0192 Joost Verhofstad 15-MAR-1983
Add JATR\$C_SESSID

V03-030 LY0299 Larry Yetto 11-FEB-1983
Increase CJF\$C_MAXATTR and CJF\$C_MAXBUFSIZ constants

V03-029 MKL044 Mary Kay Lyons 08-FEB-1983
Add create new version item list codes.

V03-028 JSV0137 Joost Verhofstad 03-FEB-1983
replace source, put in null packet

V03-027 LY0260 Larry Yetto 11-Jan-1983
Increase CJF\$C_RUFIMPSIZ to 3000. Replace JFCB\$C_MASK
which accidentally disappeared.

V03-026 LY0244 Larry Yetto 30-Dec-1982

Move RUS\$ structure from jnldefint to here. Change JFCB\$C PROCNAME to JFCB\$C SESSID. Remove JFCB\$C BINARY, PROCNODE, PROCGROUP, and PROCRUNTIME. Add Length constants to JATR structure.

V03-025 JSV0104 Joost Verhofstad 10-Dec-1982
Add CJF\$C RESET, CJF\$C ABORT, CJF\$C COMPLETED,
CJF\$C PHASE1, CJF\$C PHASE2, CJF\$C MARK,
CJF\$C RESIDUAL, CJF\$C CLEANUP

V03-024 PRB0074 Paul Beck 29-Nov-1982
Add RRP\$W_LOG_UNIT

V03-023 LY0224 Larry Yetto 12-Nov-1982
Add CJF\$C_MAX_DATA_AREA and change CJF\$C_MAX_STAGE
from 25 to 15

V03-022 PRB0042 Paul Beck 02-Nov-1982
Add CJF\$V_MERGE and CJF\$V_LOG for \$RECOVER service.

V03-021 PRB0034 Paul Beck 01-Nov-1982
Add RCP call-back definitions

V03-020 JSV0086 Joost Verhofstad 28-Oct-1982
Add CJF\$C_MAXTBUSZ

V03-019 PRB0032 Paul Beck 26-OCT-1982
Add offsets for Recovery Routine call arguments.

V03-018 PRB0029 Paul Beck 21-OCT-1982
Change recovery routine call codes from JNL\$ to
RCP\$K... and add codes for NOPR_ENTRY and
LOG_OBJECT. Also change order in RRP to create an ARB.

V03-017 JSV0083 Joost Verhofstad 18-OCT-1982
Add CJF\$C_RUFIMPSIZ

V03-016 JSV0082 Joost Verhofstad 8-OCT-1982
Add JATR\$S_ENTPRUIC (+ \$C...)

V03-015 PRB0017 Paul Beck 7-OCT-1982
Add codes for recovery routine calls (JNL\$_START etc.)

V03-014 PRB0014 Paul Beck 5-OCT-1982
Change origin of RODBA definitions, and clean up SDL
usage ("unsigned" + technique for flag arrays)

V03-013 PRB0009 Paul Beck 16-SEP-1982
Add JEN structure.

V03-012 PRB0007 Paul Beck 20-AUG-1982
Add OUTRANGE filter.

V03-011 GJA0019 Greg Awdziewicz, 19-Aug-1982 15:30
Add write modifier definitions, WRMODDEF.

(* V03-010 LY0104 Larry Yetto 16-Aug-1982
(* Add JATR\$S_BIODETA constant
(*
(* V03-09 PRB0006 Paul Beck 11-AUG-1982
(* Add MIN_VAL for JATR and JFCB.
(* V03-08 PRB0005 Paul Beck 2-AUG-1982
(* Fix position of MAX_VAL in JFCB and JATR and add warning
(* message. Add MARKPT attribute. Delete RRP\$A JEN.
(* V03-07 JSV0033 Joost Verhofstad 28-Jul-1982
(* Add JSB\$C_CL and CJF\$C_SGBSTART
(* V03-06 PRB0004 Paul Beck 20-JUL-1982
(* Add CJF\$M_RESET,COMPLETED
(* V03-05 PRB0003 Paul Beck 20-JUL-1982
(* Add RUID attribute to JATR.
(* V03-004 PRB0002 Paul Beck 16-JUL-1982
(* Add WRFLG definitions, plus ENTADR filter & attribute.
(* V03-003 PRB0001 Paul Beck 23-JUN-1982
(* Add new filters for journal name, object ID;
(* Make JFCB and JATR ranges disjoint;
(* Add RRP\$M RECOVERY UNIT to RRP\$W_FLAGS;
(* Correct RODB object types for Recovery Unit processing;
(* New symbol CJF\$C_MAXZFBUF.
(* V03-002 LY0012 Larry Yetto 18-Jun-1982
(* Change JSB\$V_SITE to JSB\$V_KNOWN
(*--

```

module %CJFDEF;          /*
/****
/*
/* CJF-flags, specified to CJF services as parameters
/* and CJF constants used internally in CJF and in the user interface
/*
/*--

aggregate %CJFDEF structure prefix CJFS;

(
( *****
( Note: we have three separate definitions of journal types:
( CJFS_XX, DTS_XXJNL, and JSBSC_XX.      Yecch.
( *****
(
  constant (
    RU          /* RU journal
    ,BI         /* BI journal
    ,AI         /* AI journal
    ,AT         /* AT journal
    ,CL         /* CL journal
    ) equals 1 increment 1 tag "":

#DTS_RUJNL = CJFS_RU;      ( Define local constants to avoid redefinition
#DTS_BIJNL = CJFS_BI;
#DTS_AIJNL = CJFS_AI;
#DTS_ATJNL = CJFS_AT;
#DTS_CLJNL = CJFS_CL;

CJFMASKS structure;      /* define CJF masks

  PHASE1        bitfield mask; /* do phase 1
  PHASE2        bitfield mask; /* do phase 2
  MARK          bitfield mask; /* make mark point
  FORWARD       bitfield mask; /* read in fifo order
  BACKWARD      bitfield mask; /* read in lifo order
  ABORT         bitfield mask; /* abort RU
                /* Note: all previous bits must go
                /* in first byte
  READ          bitfield mask; /* read indicator
  WRITE         bitfield mask; /* write indicator
  DELFIL        bitfield mask; /* delete file flag
  CONT          bitfield mask; /* continue: used with MNTJMD
  INIT          bitfield mask; /* initialize flag: with MNTJMD
  DRVWT         bitfield mask; /* driver-wait flag
  UNIT          bitfield mask; /* only specified units to be effected
  NOUNLOAD      bitfield mask; /* no-unload of medium/media
  SUPERSEDE     bitfield mask; /* supersede flag
                /* Note: all previous bits must go
                /* in first word
  RESIDUAL      bitfield mask; /* This is a residual entry indicator
  SAVFIL        bitfield mask; /* save file flag
  DISCONNECT    bitfield mask; /* Disconnect label/uic pair
  NOLOOKUP      bitfield mask; /* Do not perform known journal
                /* lookup in $ASSJNL

```



```

ADDFILTER    bitfield mask; /* Add filter ($MODFLT)
DELFILTER    bitfield mask; /* Delete filter ($MODFLT)
COMPLETED   bitfield mask; /* RU completed successfully
RESET        bitfield mask; /* RU was reset to mark (ID in attributes)
REMOUNT      bitfield mask; /* Recovery for volume mount
FORCE        bitfield mask; /* Modifier for Forward/Backward
RESTART      bitfield mask; /* Restart frozen REMOUNT recovery op
FAILOVER     bitfield mask; /* Failover RUs for remastered journal
LOG          bitfield mask; /* RECOVER/LOG request
MERGE        bitfield mask; /* Merge new facility with RCP
LOGOBJ       bitfield mask; /* Request list of frozen objects
ROOTDEV      bitfield mask; /* Indicating root RU journal or not
RCP          bitfield mask; /* Service call coming from the RCP

```

end CJFMASKS;

```

constant ABENUM equals 8 tag C; /* Number of ABEs allocated per ABL
constant BUFIO MAX equals 200 tag C; /* Max size buffer for which BIO done
constant DEFBUFSIZ equals 512 tag C; /* default buffer size in bytes
constant INITADBNUM equals 8 tag C; /* initial number of ADBs per ADL
constant MAXATTR equals 27 tag C; /* Max number of attributes returned
/* with read.
constant MAXBUFSIZ equals 20 tag C; /* max buffer size in 512 b. blocks
constant MAXCOPIES equals 1 tag C; /* Max number of journal copies allowed
constant MAXFILLEN equals 255 tag C; /* max file length
constant MAXFILT equals 512 tag C; /* Max filter size
constant MAXITEMS equals 21 tag C; /* Max number of items in items list for MOUNT
constant MAXJNLS equals 30 tag C; /* max number of journals on one tape group
constant MAXAIJNL equals 31 tag C; /* max AI journals that can be accessed
constant MAXBIJNL equals 31 tag C; /* max BI journals that can be accessed
constant MAXRUJNL equals 31 tag C; /* max RU journals that can be accessed
constant MAXRECSIZ equals 32767 tag C; /* max journal entry size
constant MAXSPLFIL equals 5 tag C; /* max number of spool files
constant MAXTBUFSZ equals 2048 tag C; /* max size journal tape buffer in bytes
constant MAX_STAGE equals 15 tag C; /* max number of next stage macros in
/* one routine
constant MAX_DATA_AREA equals 15*12 tag C; /* max bytes of next stage
/* data allowed in one routine
constant MXDEVNAML equals 20 tag C; /* maximum length device name
constant MXGRPNAML equals 15 tag C; /* maximum group name length
constant MXITEMLEN equals 20 tag C; /* max length item in item list for MOUNT
constant MXJNLNAML equals 16 tag C; /* max length journal name
constant MXLENATR equals 20 tag C; /* Max length attribute field
constant MXLENFIL equals 64 tag C; /* Max length filter element
constant MXPRCNAML equals 15 tag C; /* Max length process name string
constant MXSGBLEN equals 255 tag C; /* Max SGB field length
constant NUMVLE equals 8 tag C; /* initial and incremental number of VLEs
constant RDBUFMAX equals 512 tag C; /* Max buffer that can be used for reading
constant RUFIMPSIZ equals 3000 tag C; /* RUF impure area size
constant RULINC equals 5 tag C; /* RUL increment: number of RUEs added
/* when RUL is full
constant RUDLEN equals 1 tag C; /* Starting number of RUEs in RUL
constant SGBSTART equals 128 tag C; /* Starting number for SGB codes
constant ACPUIC equals ((1016) + 3) tag C; /* ACP's UIC ([1,3])
constant MAXZFBUF equals 127 tag C; /* Max size buf for 0-filling jnl file

```



```

/**
/*
/* Definitions for codes indicating the type of RU-control entry.
/*
/*-
  constant
    (
      PHASE1      /* phase1 entry
    ,PHASE2      /* phase2 entry
    ,ABORT        /* abort entry
    ,MARK         /* mark point entry
    ,RESET        /* reset entry
    ,COMPLETED   /* completed entry
    ,RESIDUAL     /* residual entry
    ,CLEANUP      /* cleanup entry
    ) equals 1 increment 1 prefix CJFS tag "C";

/**
/*
/* Definitions for Recovery Control Process and Recovery Routines
/*
/* Miscellaneous constants
/*-
  constant    LOGLENGTH equals 512 prefix RCP$; /* max length of log mbx msg
/**
/* Offsets to argument list supplied to Recovery Routines by the RCP.
/*-
  constant
    (
      COMMAND      /* Command code defines type of call (see below)
    ,RRP           /* Address of Recovery Request Packet
    ,JEN           /* Address of Journal Entry
    ,IMPURE        /* Address of RR-supplied impure data area
    ,ASTADR        /* Address of RCP-supplied AST for asynch op's
    ,CALLBAK       /* Address of RCP-supplied callback routine
    ) equals 4 increment 4 prefix RCPARG$ tag "...";

/**
/* Command codes for Recovery Routines within Recovery Control Process
/* (RCPARG$_COMMAND).
/*-
  constant
    (
      START        /* Start of Recovery Operation
    ,PROCESS       /* Process a Journal Entry
    ,END           /* End of Recovery Operation
    ,MAP_ENTRY     /* Process Journal Entry as Mapping Entry
    ,LOCK_ENTRY    /* Lock a Journal Entry
    ,ABORT         /* Abort a Recovery Operation
    ,NOPR_ENTRY    /* Process Journal Entry; caller lacks privs
    ,LOG_OBJECT    /* Return log information about OBJECT ID entry
    ,DIRECTION     /* Request direction to roll Phase 1 RO
    ,LASTPLUS1     /* *** Insert new entries before this ***
    ) equals 1 increment 1 prefix RCP$ ;

  #max_command = RCP$K_LASTPLUS1;
  constant    MAX_COMMAND equals #max_command-1 prefix RCP$; /* Adjust back
/**
/* Codes to pass as P1 to RCPARG$_CALLBAK (RCP callback routine).
/*-

```

```
constant (
    WAIT      /* Wait for AST (no additional arguments)
    .FADD      /* Add Filter (P2=Address of filter list)
    .FDEL      /* Delete Filter (P2=Address of filter list)
    .LOGMSG    /* Log Message (P2=Address of message descr.)
    .MESSAG    /* Other message (P2=Addr of msg descr.)
) equals 64 increment 1 prefix RCP$ ;

end      SCJFDEF;
end_module;
```



```
module $WRFLGDEF;
/*++
/*
/* WRFLG - Flags supplied with $WRITEJNL
/*
/* These flags may be supplied with a call to $WRITEJNL to accompany the
/* journal entry. The flags may later be read as an attribute (JATR$C_ENTATR) of
/* the journal entry via $READJNL or may be used as a filter (JFCB$C_ENTATR) to
/* $POSJNL to select flagged journal entries.
/*
/*--
aggregate $WRFLGDEF structure prefix WRFLG$;

  WRFLG structure;                /* define CJF masks

    AI      bitfield mask;        /* This is a roll forward (AI) entry
    BI      bitfield mask;        /* This is a roll back (BI) entry
    RUALSO  bitfield mask;        /* This AI/BI entry also written to RUJ
    OBJECT_ID bitfield mask;      /* This entry identifies an object
    LOCK    bitfield mask;        /* This entry contains locking info

  end WRFLG;

end $WRFLGDEF;
end_module;
```

```
module $WRMODDEF;
/*++
/*
/* WRMOD - Modifiers supplied with $WRITEJNL
/*
/* These flags may be supplied with a call to $WRITEJNL to modify the write
/* QIO. They are defined to have the same values as the corresponding IO
/* modifiers.
/*
/*--
aggregate $WRMODDEF structure prefix WRMOD$:
    WRMOD structure;                /* define write modifier masks
        foo    bitfield length 6 fill; /* Skip 6 bit positions so that the
        FORCE    bitfield mask;        /* defined values agree with IOSDEF.
        CNTRLENTY bitfield mask;      /* force entry to be written to file.
        /* This is a control entry.
    end WRMOD;
end $WRMODDEF;
end_module;
```



```
module $ILEDEF;      /*
/*++
/*
/* ILE - Item List Element for MOUNT
/*
/* The item list parameter for MOUNT consists of these items
/*
/*--
aggregate $ILEDEF structure prefix ILES;

    BUFLN      word unsigned; /* buffer length
    ITEMCODE   word unsigned; /* item code
    BUFFADDR   longword unsigned; /* buffer address
    RESLEN     longword unsigned; /* result length (used for GETDVI)

    constant "LENGTH" equals .; /* length of data structure
    constant "LENGTH" equals . tag C; /* length of data structure

end      $ILEDEF;
end_module;
```

```

module $JATRDEF;      /*
/**
/*
/* JATR - Journal Attributes
/*
/* A READ-Journal operation ($READJNL) can also be used to get attributes
/* of the entry being read and/or attributes of the journal from which
/* reading is done. The attribute block is a vector of attribute descriptors
/* which contain attribute type and attribute size in the first longword
/* and the address of the user buffer for the attribute in the second longword.
/* The attribute block is zero ended.
/*
/* ***** WARNING *****
/*   If any new attributes are added or old attributes removed make sure that
/*   the constant CJFSC_MAXATTR is updated to reflect the change.
/* ***** WARNING *****
/*--

aggregate $JATRDEF structure prefix JATR$:

    SIZE      word unsigned; /* size of attribute descriptor block
    TYPE      word unsigned; /* type of attribute
    ADDR      longword unsigned; /* address user buffer for attribute

    constant CTRLBLCKSIZ equals . tag C; /* control block size
    constant 'LENGTH' equals . tag C;
    constant 'LENGTH' equals .;

#JATR_BASE = 3;
    constant MIN_VAL equals #JATR_BASE tag C; /* define low limit
/**
/* Attribute codes
/* ***** WARNING *****
/*   If any new attributes are added or old attributes removed make sure that
/*   the constant CJFSC_MAXATTR is updated to reflect the change.
/* ***** WARNING *****
/*--
    constant (
        TIME          /* time
        .ENTMOD        /* access mode of entry
        .JNLMOD        /* access mode for journal
        .SEQNO         /* sequence number of entry
        .ENTMASK       /* mask given to entry at write
        .JNLMASK       /* journal mask
        .JNLCDAT       /* journal device creation date/time
        .FILCDAT       /* journal file creation date/time
        .COPAVL        /* number of journal copies available
        .COPEXI        /* number of journal copies existing
        .ENTUIC        /* UIC of entry
        .JNLUIC        /* UIC of journal
        .ENTPROT       /* protection mask of entry
        .JNLPROT       /* protection mask of journal
        .FILSIZ        /* journal file size (disk journals only)
        .BIODATA       /* BIO journal entry - internal only
        .DIODATA       /* DIO journal entry - internal only
        .ENTLEN        /* entry length

```



```

,XFERCNT      /* Count of # of bytes actually transfered.
,FACCOD       /* facility code
,MAXENTSIZ    /* Maximum size (bytes) of a journal entry.
,ENTATR       /* Journal entry attribute flags (WRFLGS...)
,RUID         /* Recovery Unit ID for this entry
,MARKPT       /* Markpoint ID for this entry
,ENTPRUIC     /* UIC of process that wrote entry
,SESSID       /* Session ID from which entry was written
,BUFSIZ       /* Size of buffers for journal in 512 byte blocks
,MAX_VAL      /* maximum value * MUST BE LAST ENTRY IN LIST *
) equals #JATR_BASE increment 1 tag C;
/* ***** WARNING *****
/*   If any new attributes are added or old attributes removed make sure that
/*   the constant CJFSC_MAXATTR is updated to reflect the change.
/* ***** WARNING *****

constant      TIME      equals 8 tag S; /* attribute sizes
constant      ENTMOD     equals 1 tag S; /* time
constant      JNLMOD     equals 1 tag S; /* access mode of entry
constant      SEQNO      equals 4 tag S; /* access mode for journal
constant      ENTMASK     equals 4 tag S; /* sequence number of entry
constant      JNLMASK     equals 4 tag S; /* mask given to entry at write
constant      JNLCDAT     equals 8 tag S; /* journal mask
constant      FILCRDAT    equals 8 tag S; /* journal device creation date/time
constant      COPAVL      equals 1 tag S; /* journal file creation date/time
constant      COPEXI      equals 1 tag S; /* number of journal copies available
constant      ENTUIC      equals 4 tag S; /* number of journal copies existing
constant      JNLUIC      equals 4 tag S; /* UIC of entry
constant      ENTPROT     equals 2 tag S; /* UIC of journal
constant      JNLPROT     equals 2 tag S; /* protection mask of entry
constant      FILSIZ      equals 4 tag S; /* protection mask of journal
constant      BIODATA     equals 8 tag S; /* journal file size (disk journals only)
constant      DIODATA     equals 8 tag S; /* BIO journal entry - internal only
constant      ENTLEN      equals 4 tag S; /* DIO journal entry - internal only
constant      XFERCNT     equals 4 tag S; /* entry length
constant      FACCOD      equals 2 tag S; /* Transfer count (bytes).
constant      MAXENTSIZ   equals 2 tag S; /* facility code
constant      ENTATR      equals 4 tag S; /* Maximum size (bytes) of a journal entry.
constant      RUID        equals 16 tag S; /* Journal entry attribute flag
constant      MARKPT      equals 4 tag S; /* Recovery Unit ID
constant      ENTPRUIC    equals 4 tag S; /* Mark point ID
constant      SESSID      equals 16 tag S; /* UIC of process that wrote entry
constant      BUFSIZ      equals 2 tag S; /* Session ID from which entry was written
constant      MAX_VAL     equals 1 tag S; /* Buffer size in 512 byte blocks
/* ***** WARNING *****
/*   If any new attributes are added or old attributes removed make sure that
/*   the constant CJFSC_MAXATTR is updated to reflect the change.
/* ***** WARNING *****

end      $JATRDEF;

end_module;

```

```

module $JFCBDEF; /*
/++
/*
/* JFCB - Journal Filter Control Block
/*
/* When a POSJNL (Position Journal) is done, the caller specifies the
/* category of entries he wants to read. This is done by passing a filter.
/* The filter consists of a vector of descriptors, which have in their
/* first longword the size and type of the filter element, and in their
/* second longword the address of the user buffer containing the value of
/* the filter element
/*
/* *** NOTE ESPECIALLY: The RUID and MARK filters are valid only for RU journals
/* and are the ONLY filters used for RU journals; furthermore,
/* the RUID filter MUST be provided for RU journals.
/*
/+-
aggregate $JFCBDEF structure prefix JFCBS;

SIZE      word unsigned; /* size of filter element descriptor block
TYPE      word unsigned; /* type of filter element
ADDR      longword unsigned; /* address user buffer for filter element

constant CTRLBLCKSIZ equals . tag C; /* control block size
constant "LENGTH" equals . tag C;
constant "LENGTH" equals .;

TERM      longword unsigned; /* offset to terminating zero from last entry
/++
/* Filter element codes
/*
#JFCB_BASE = 64;
constant MIN_VAL equals #JFCB_BASE tag C; /* define low limit

constant (
    UIC          /* UIC of write. of entry
    .ACMODE      /* access mode from which entry was written
    .FACCODE     /* facility code of facility that wrote entry
    .STRING      /* field describing: in first word offset
                  /* in user entry, in rest of field
                  /* string to match entry's subfield
    .MASK        /* mask given to entry at write
    .SEQNO       /* sequence number or range: lowest # for
                  /* reading AT and BI jnl's, highest # for
                  /* reading AT and AI jnl's
                  /* if two longwords supplied, range of sequence #s
    .DATIM       /* date time up to and/or from which to read
    .RUID        /* RU identifier
    .MARK        /* MARK point ID, up to which to roll back
    .SESSID      /* session ID
    .PID         /* Process ID
    .ACTIVE      /* Describes what action to take with
                  /* active processes
    .spare       /* *** put next new filter here ***
    .JOURNAL     /* Journal name (for RCP - not used by ACP)

```



```

,ENTATR      /* Journal entry attribute flags
,OUTRANGE    /* matches entries outside range of seq #s
,MAX_VAL     /* maximum value * MUST BE LAST ENTRY IN LIST *
) equals #JFCB_BASE increment 1 tag C;

```

```

constant UIC      equals 4   tag S; /* UIC of writer of entry
constant ACMODE   equals 1   tag S; /* access mode from which entry was written
constant FACCODE  equals 2   tag S; /* facility code of of facility that wrote entry
constant STRING   equals 512 tag S; /* field describing: in first word offset
                                   /* in user entry in second word length
                                   /* of subfield, in rest of field
                                   /* string to match entry's subfield
constant 'MASK'   equals 4   tag S; /* mask given to entry at write
constant SEQNO    equals 8   tag S; /* sequence number(s)
constant DATTIM   equals 16  tag S; /* date time up to and/or from which to read
constant RUID     equals 16  tag S; /* RU identifier
constant MARK     equals 4   tag S; /* MARK point ID, up to which to roll back
constant SESSID   equals 15  tag S; /* session ID
constant PID      equals 4   tag S; /* process ID
constant ACTIVE   equals 1   tag S; /* Active process flag
constant JOURNAL  equals 12  tag S; /* journal name
constant ENTATR   equals 4   tag S; /* Journal entry attribute flag
constant OUTRANGE equals 8   tag S; /* low, high limit sequence numbers
constant MAX_VAL  equals 4   tag S; /* maximum value

```

```

constant (
  EXCLUDE /* Exclude entries from active processes
  ,INCLUDE /* Include entries from active processes
  ,NONE    /* Ignore effects of active processes
) equals 0 increment 1 tag C;

```

```

/* Special INTERNAL definitions of the JFCB structure (used by the driver
/* and the ACP only.

```

```

constant DIRECT equals JFCBSC_MAX_VAL tag C; /* The typecode for the direction flag is
                                   /* set to the maximum type-code value.
constant DIRECT equals 4 tag S; /* The size of the flag is 4 (?) bytes.

```

```

end $JFCBDEF;

```

```

end_module;

```

```
module $JNLCHARDEF; /*
/*++
/*
/* JNLCHAR - journal characteristics bits
/*
/*--
aggregate $JNLCHARDEF structure prefix JNLCHAR$:

    RESWL      bitfield mask;      /* reset SWL for jnl
    SEAVL      bitfield mask;      /* set def on-line again
    REAVL      bitfield mask;      /* take device off-line

end $JNLCHARDEF;
end_module;
```

```
module $JSBDEF;      /*
```

```
/*++
```

```
/*
```

```
/* JSB - Journal Specification Block
```

```
/*
```

```
/* When a journal is to be created, the user must pass this
```

```
/* structure to the CJF $CREJNL service.
```

```
/*
```

```
/*--
```

```
aggregate $JSBDEF structure prefix JSB$;
```

```
JNLNAMLEN word unsigned; /* length journal name
```

```
SPARE0 word unsigned fill; /* spare
```

```
JNLNAM longword unsigned; /* journal name ASCII
```

```
SPARE word unsigned fill; /* spare
```

```
JNLTP byte unsigned; /* journal type. can be one of:
```

```
constant RU equals #DTS_RUJNL tag C; /* RU journal
```

```
constant BI equals #DTS_BIJNL tag C; /* BI journal
```

```
constant AI equals #DTS_AIJNL tag C; /* AI journal
```

```
constant AT equals #DTS_ATJNL tag C; /* AT journal
```

```
constant CL equals #DTS_CLJNL tag C; /* CL journal
```

```
JNLDEV byte unsigned; /* journal device type. can be one of:
```

```
constant {
```

```
DISK /* journal is on disk
```

```
TAPE /* journal is on tape
```

```
} equals 1 increment 1 tag C;
```

```
'MASK' longword unsigned; /* journal mask
```

```
FACCOD word unsigned; /* facility code (eg RMS)
```

```
APPLID word unsigned; /* applications id (eg datatrieve)
```

```
MAXSIZ word unsigned; /* maximum entry size
```

```
spare1 word unsigned; /* spare
```

```
FILSIZ longword unsigned; /* blocks to initially allocate for journal file
```

```
FILEXT word unsigned; /* blocks to extend journal file when full
```

```
BUFSIZ word unsigned; /* buffer size (in 512 byte blocks)
```

```
QUOTA longword unsigned; /* byte quota (for RU journals only)
```

```
ACMODE byte unsigned; /* least priv access mode allowed
```

```
constant {
```

```
KERNEL /* kernel mode
```

```
.EXEC /* exec mode
```

```
.SUPER /* supervisor mode
```

```
.USER /* user mode
```

```
) equals 0 increment 1 tag C;
```

```
spare2 byte unsigned; /* spare (for longword alignment)
```

```
PROT word unsigned; /* protection mask for journal device
```

```
uic_overlay union;
```

```
UIC longword unsigned; /* UIC for journal device
```

```
uic_0 structure; /*
```



```

    UIC_MBM word unsigned; /* UIC member number
    UIC_GRP word unsigned; /* UIC group number
end_uic_0;
end_uic_overlay;

flags_overlay union;
  FLAGS longword unsigned; /* flags as follows:
  flags_bits structure; /*
    /* NOTE: flags are used in prologue -
    /* must be in same places.
    TMPJNL bitfield mask; /* temporary journal device - delete on last deaccess
    KNOWN bitfield mask; /* site permanent journal
    CREATE bitfield mask; /* always create file (supercedes CREATE_IF)
    CIF bitfield mask; /* create only if file does not exist
    TMPFIL bitfield mask; /* temporary journal file - delete when device deleted
    CREACP bitfield mask; /* create a new ACP. OPER priv req'd.
    /* ACP name in JSB is valid.
    DIFACP bitfield mask; /* do not use default ACP.
    /* ACP name in JSB is valid.
    REPLACE bitfield mask; /* replace current journal with this
    /* DELETE priv required
    TAPE_DRIVE bitfield mask; /* (internal only) create journal tape drive
    CLUSTER bitfield mask; /* create the journal across
    /* the cluster
    REMASTER bitfield mask; /* (internal only) remaster the journal
    filler bitfield length 32-^ fill;
end_uic_flags_bits;
end_uic_flags_overlay;

ACPNAMLEN word unsigned; /* length prcnam string
SPARE4 word unsigned; /* spare
ACPNAM longword unsigned; /* prcnam of alternate ACP

MAX_JNLS word unsigned; /* (internal only) max jnls (if TAPE_DRIVE set)
COPIES byte unsigned; /* number of copies
SPARE3 byte unsigned; /* SPARE (for longword alignment)

EXPDAT quadword unsigned; /* expiration date (-1 = never)

/* Primary file specifications
PRINAMDES longword unsigned; /* address of filename descriptor list
/* (one quadword per file) (required)
PRIRESDES longword unsigned; /* address of result string descriptor list
/* (one quadword per file) (optional)
PRIRESLEN longword unsigned; /* address of result length list
/* (one longword per file) (optional)

constant 'LENGTH' equals .; /* length of data structure
constant 'LENGTH' equals . tag C; /* length of data structure

end $JSBDEF;
end_module;

```

```

module $RODBDEF;      /*
/****
/*
/* RODB - Recovery Object Descriptor Block
/*
/* The RODB describes the object to recover.
/*
/*--
aggregate $RODBDEF structure prefix RODBS;

    TYPE          byte unsigned; /* Type of object described
    constant      (
        RMSFILE    /* ...Object is an RMS file
        .VOLUME    /* ...Object is a volume set
        .RU        /* ...Object is a Recovery Unit
        .RUJNL     /* ...Object is a Recovery Unit Journal
        .spare     /* ...RUNODE removed from here
        .FACCOD    /* ...Facility code for use with merge ** INTERNAL USE ONLY **
        .RFSAMPLE  /* ...Object is the Sample Application
    ) equals 1 increment 1;
    COUNT          byte unsigned; /* Number of attributes assoc with object
    SIZE           word unsigned; /* Size of RODBA
    POINTER        address;       /* Address of attribute list

    constant 'LENGTH' equals .; /* Length of this structure
    constant 'LENGTH' equals . tag C; /* length of data structure

end $RODBDEF;
/*
/****
/*
/* RODBA - Recovery Object Descriptor Block Attribute
/*
/* The RODBA describes an attribute associated with the object of recovery.
/*
/*--
aggregate $RODBADEF structure prefix RODBAS;

    TYPE          byte unsigned; /* Type of object described
    spare         byte unsigned; /* Spare
    SIZE          word unsigned; /* Size of attribute
    POINTER       address;       /* Pointer to attribute

    constant 'LENGTH' equals .; /* Length of this structure
    constant 'LENGTH' equals . tag C; /* length of data structure
/****
/* Define attributes for RMS object attributes
/*--
    constant      (
        FILENAME    /* File name to recover
        .CFILENAME  /* File name to create and recover
/****
/* Define attributes for Volume Recovery attributes
/*--
        .VOLDEVICE  /* Volume device name
        .VOLLABEL   /* Volume label

```

```
/*  
/* Define attributes for RU attributes  
/*  
/*      ,RUID                /* Recovery Unit ID  
/*  
/* Define attributes for RUJNL attributes  
/*  
/*      ,RUJDEVNAM           /* Device name of RU journal  
/*  
/* Define attributes for failed node (PROCESSOR) attributes  
/*  
/*      ,NODE_ID             /* Node ID of failed node  
/*  
/* Define attribute for facility code to be used with merge command.  
/*  
/*      ,FACCOD              /* Facility code *** INTERNAL USE ONLY ***  
/*  
/* End of definitions  
/*  
/*      ) equals 1 increment 1;  
  
end $RODBADEF;  
end_module;
```



```

module $RRPDEF;      /*
/****
/*
/* RRP - Recovery Request Packet
/*
/* When a recovery is to be performed a Recovery Request Packet must
/* be sent to the RCP that describes the caller, the object to be
/* recovered and the type of recovery.
/*
/*--
aggregate $RRPDEF structure prefix RRP$;

FLINK      address;      /* Forward Link
BLINK      address;      /* Backward Link
SIZE       word unsigned; /* Actual allocation size (in RCP)
TYPE       byte unsigned; /* Structure type
SUBTYPE    byte unsigned; /* Structure subtype

flags overlay union;
  FLAGS     word unsigned; /* Request flags:
  flags_bits structure;    /*
                        /* These two flags are exclusive
  RECOVERY  bitfield mask; /* If set, start Recovery Operation
  MERGE     bitfield mask; /* If set, merge in new facility
                        /* These flags apply only to RECOVERY
  FORWARD  bitfield mask; /* Roll-forward operation
  BACK     bitfield mask; /* Roll-back operation
  RECOVERY UNIT bitfield mask; /* Process a Recovery Unit
  REMOUNT  bitfield mask; /* Process a Recovery Unit Journal
  FORCE     bitfield mask; /* Process a frozen Recovery Unit
                        /* ..for BI, roll back over RUALSO entries for
                        /* ..successful RUS.
  LOG      bitfield mask; /* Log recovery events
  FAILOVER bitfield mask; /* Failover of RUs from failed node in cluster
  RESTART  bitfield mask; /* Restart of frozen REMOUNT
  filler   bitfield length 16-^ fill;
end        flags_bits;
end        flags_overlay;

FACNO      word unsigned; /* Facility number (RRPSM_MERGE only)
TIME       quadword unsigned; /* Time of request
CALL_PRIV  quadword unsigned; /* Privilege mask of CALLER
CALL_PID   longword unsigned; /* CALLER's process ID (EPID form)
CALL_UIC   longword unsigned; /* UIC of calling process
CALL_NODE  longword unsigned; /* CSID of CALLER
CALL_AMOD  byte unsigned; /* Access mode of CALLER
CALL_MSG   byte unsigned; /* CTL$GB_MSGMASK of CALLER
FILTER     address;      /* Address of filter descr part of RRP
RODB       address;      /* Address of RODB descr part of RRP
constant MBX_SIZE equals 512; /* Size of status MBX to create
MBX_UNIT   word unsigned; /* Status MBX unit number
LOG_UNIT   word unsigned; /* Logging MBX unit number
constant FIXED equals .; /* Size of fixed portion of RRP
WORK       byte tag AB; /* Start of RODBs, filters, and Journals

end $RRPDEF;

```

end_module;

```

module $RUSDEF;      /*

/****
/*
/* RUS - List of recovery units as returned from IOS_RUCNTR ! IOSM_RUIDLIST OR ! RUJLIST
/*
/* This structure is used to return the list of recovery units outstanding
/* in a recovery unit to which the RUCNTR operation is done. This is an
/* internal QIO - not available to users, so this data structure is for
/* internal purposes only.
/*
/* NOTE that the status bit definitions must be the same as those for RUE.
/*
/*--

aggregate $RUSDEF structure prefix RU$:

  RUID          octaword unsigned ;    /* Recovery Unit ID
  SEQNO         longword unsigned ;    /* sequence number last entry written
  JNLCNT        word unsigned ;        /* count of journals touched by RU
  spare1        word unsigned ;        /* Spare word to keep longword boundary
  INDEX         longword unsigned ;    /* unique short RUE index

  status_overlay union ;
  STATUS        longword unsigned ;    /* status of the Recovery Unit

    status_bits structure ;
    PURGED bitfield mask;               /* entry is free indicator
    ROLL_BACK bitfield mask;            /* there is at least one roll back entry
    ROLL_FORW bitfield mask;            /* there is at least one roll forward entry
    NOT_FLSHD bitfield mask;            /* there is at least one entry not flushed
    OVER_QUOTA bitfield mask;           /* quota exceeded
    PHASE1 bitfield mask;               /* phase1 done
    PHASE2 bitfield mask;               /* phase2 done
    ABORT bitfield mask;                /* abort done
    P2$AB$2 bitfield mask;              /* phase2 or abort entry to be encountered 2*
                                         /* before RU deletion
    RESIDUAL bitfield mask;              /* this is a residual RU in journal
    COMPLETED bitfield mask;            /* RU has been completed (rolled forward)
    CLEANUP bitfield mask;              /* vestigial entry for RU can be ignored
    FROZEN bitfield mask;               /* frozen RU
    RUSYNCEX bitfield mask;             /* RUSYNC entry expected
    RUSYNCR bitfield mask;             /* RUSYNC entry written
    NOFAC bitfield mask;                /* Frozen due to missing facility
    NOOBJ bitfield mask;                /* Frozen due to missing object
    filler      bitfield length 32-^ fill ;
  end status_bits ;
end status_overlay ;

  DEVNAM        character length 16 ;    /* Counted ASCII device name
  constant "LENGTH" equals ;            /* length of structure
  constant "LENGTH" equals : tag C ;    /* length of structure

end $RUSDEF;

```


JNLDEF.SDL;1

16-SEP-1984 16:40:00.43⁹ Page 24

end_module;

```

module $JENDEF;
/*
/* JEN - Journal Entry
/*
/* Contains a pointer to the journal entry, plus related attributes as
/* returned by the JACP and passed to the recovery routines by the RCP.
/*
aggregate $JENDEF structure prefix JENS;

    FLINK      address;      /* Forward link
    BLINK      address;      /* Backward link
    TIME       quadword unsigned; /* System time of journal entry
    FACNO      word unsigned; /* Facility number of writer of journal entry
    DIRECTION  byte unsigned; /* Direction of recovery
    TYPE       byte unsigned; /* Type of journal (DTS...)
    SEQNO      longword unsigned; /* Sequence number of journal entry
    ENTMOD     byte unsigned; /* Access mode of journal entry
    JNLMOD     byte unsigned; /* Access mode of journal
    ENTUIC     longword unsigned; /* UIC of journal entry
    JNLUIC     longword unsigned; /* UIC of journal
    ENTPROT    word unsigned; /* Protection mask of journal entry
    JNLPROT    word unsigned; /* Protection mask of journal
    ENTATR     longword unsigned; /* Journal entry attribute flags
    ruidblock  structure; /* Recovery Unit ID
        RUID    byte unsigned dimension 16 tag 0;
    end
    MARKPT     longword unsigned; /* MARKpointID for MARK/RESET control entries
    CHANNEL    word unsigned; /* Channel assigned to journal
    JNLNAME    character length 13; /* Journal name, counted ASCII string
    ENTSIZE    word unsigned; /* Length of journal entry in bytes
    'ENTRY'    address; /* Address of Journal Entry buffer
    IOSB_DATA  longword unsigned; /* copy of JLE second longword (RCP use only)

    constant 'LENGTH' equals .; /* length of data structure

end $JENDEF;
end_module;

```

```

module $SGBDEF;      /*
/****
/*
/* SGB - Shadow Group Block
/*
/* When a shadow group is mounted the user must pass this
/* structure to the CJF $MNTJMD service. This is a list of
/* descriptors like the item list, which must be zero ended.
/*
/*--

aggregate $SGBBIT structure prefix SGB$;
    INIT          bitfield mask; /* bits defined for FLAGS
end $SGBBIT;

aggregate $SGBDEF structure prefix SGB$;
    SIZE          word unsigned; /* size of field descriptor block
    TYPE          word unsigned; /* type of SGB field
    ADDR          longword unsigned; /* address user buffer for attribute
    SPARE         longword unsigned; /* spare
/****
/* SGB codes
/*--
    constant (
        GRPNAME          /* group logical name
        ,COPIES          /* # of spool files
        ,PROT            /* protection mask for shadow group
        ,MAX_JNLS        /* max journals allowed in shadow group
        ,FLAGS           /* flags
        ,UIC             /* uic for shadow group
        ,SPLFILSIZ       /* blocks to allocate for spool file
        ,SPLNAM          /* Spool file name
        ,SPLRESNAM       /* Result spool file name
        ,SPLRESLEN       /* Result spool file name length
        ,MAX_VAL
    ) equals 255 increment -1 tag C;

    constant 'LENGTH' equals .; /* Length of this structure
    constant 'LENGTH' equals . tag C; /* length of data structure

end $SGBDEF;
end_module;

```



```
module $CNVDEF;
```

```
/*
```

```
/* Definitions for the create new version item list codes
```

```
*/
```

```
aggregate CNVDEF union prefix CNV$;
```

```
  CNVDEF BITS structure;
```

```
    CLOSE bitfield mask;
```

```
  end CNVDEF_BITS;
```

```
  #min = 1;
```

```
  constant
```

```
    (MIN_VAL) equals #min prefix CNV tag '$'
```

```
    , (CURDEVNAM) /* Current journal copy device name
```

```
    , (NEWDEVNAM) /* Device name for new version of journal copy
```

```
    , (NEWVER) /* Version number for new version of copy
```

```
    , (FILDEVNAM) /* Device name for copy to connect/disconnect
```

```
    , (FILNAM) /* File name for copy to connect/disconnect
```

```
    , (FILVER) /* Version number for copy to connect/disconnect
```

```
    , (OLDFILNAM) /* File name for create new version rename
```

```
    , (ALQ) /* Allocation quantity for new version
```

```
    , (FLAGS) /* Flags
```

```
    , (IOSB) /* Secondary status from create new version
```

```
    , (NVRSA) /* Result buffer address for new version spec.
```

```
    , (NVRSL) /* Result buffer size for new version file spec.
```

```
    , (OVRSA) /* Result buffer address for old version spec.
```

```
    , (OVRSL) /* Result buffer size for old version file spec.
```

```
    ) equals #min increment 1 counter #cnvctr prefix CNV tag '$'
```

```
    , (MAX_VAL) equals #cnvctr prefix CNV tag '$';
```

```
end CNVDEF;
```

```
end_module $CNVDEF;
```

```
module $CJDEF;  
/*  
/* Definitions for the $GETCJI service item codes  
/*  
aggregate CJDEF union prefix CJI$;  
    #min = 1;  
    constant (MIN_VAL) equals #min prefix CJI tag '$'  
              (FICDSKNAM /* Get the journal file disk name  
              ) equals #min increment 1 counter #cjictr prefix CJI tag '$'  
              (MAX_VAL) equals #cjictr prefix CJI tag '$';  
end CJDEF;  
end_module $CJDEF;
```


0045 AH-BT13A-SE
VAX/VMS V4.0

**DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY**